Amendments to the Claims

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (previously presented) A computer-implemented method for automatically invoking at least one predetermined debugger command at a desired location of a single thread of a program containing at least one thread, said method comprising:

(a) embedding within said single thread at said desired location thereof a utility which reads a trace file in which said at least one predetermined debugger command has been previously embedded; and

(b) running the program so that on reaching said desired location, the utility is invoked for reading said trace file and invoking said at least one predetermined debugger command.

B

2. (canceled)

3. (previously presented) The method according to Claim 1, wherein the debugger attaches itself to a predetermined debugger command, which halts execution of the program and shows a state of the program at that time.

4. (previously presented) The method according to Claim 1, wherein the program is multi-threaded and there is included the further operation of:

(c) providing a mechanism for rerunning the program with identical interleaving as far as instrumentation statements are concerned.

5. (previously presented) The method according to Claim 1, further including the operation of creating the mechanism automatically using the instrumentation statements.

6. (previously presented) The method according to Claim 1, wherein the program includes multiple threads, each of which prints an invariant associated with a status of the program, said invariant having a value that remains constant regardless of the interleaving.

7. (previously presented) A computer-implemented method for automatically invoking at least one predetermined debugger command at a desired location of a specific thread of a program containing at least one thread, said method comprising:

(' B

(a) embedding within the specific thread of the program at said desired location thereof a utility which:

i) checks whether a trace file exists on entry to the utility,

ii) if the trace file does not exist on entry to the utility, creates the trace file and writes a traced value of at least one variable thereto at a desired location of the program, and

iii) if the trace file does exist on entry to the utility, compares a current value of the least one variable with a respective line in the trace file and if they are different invokes a debugger so as to execute a debugger command embedded in the trace file in place of the traced value; and

(b) running the program so that on reaching said desired location, the utility is invoked for reading a modified trace file readable by the program wherein at least one traced value is replaced or augmented by said debugger command.

8. (previously presented) The method according to Claim 7, wherein the at least one predetermined debugger command halts execution of the program and shows the state of the program at that time.

9. (previously presented) The method according to Claim 7, wherein the program is multi-threaded and there is included the further operation of:

(c) providing a mechanism for rerunning the program with identical interleaving as far as instrumentation statements are concerned.

B

10. (previously presented) The method according to Claim 9, including the further operation of creating the mechanism automatically using the instrumentation statements.

11. (original) The method according to Claim 7, wherein the program includes multiple threads, each of which prints an invariant associated with a status of the program, said invariant having a value that remains constant regardless of the interleaving.

12. (previously presented) The method according to Claim 7, wherein the program includes multiple threads and operation **(a)(ii)** includes:

i) creating for each thread a respective trace file having a name which is uniquely defined by a name of the respective thread;

thereby allowing debugger commands embedded in any of the trace files to be executed during a respective one of the threads.

13. (previously presented) The method according to Claim 12, further including the operation of automatically naming said trace files according to a predetermined execution-independent naming scheme.

14. (previously presented) The method according to Claim 13, wherein said naming scheme includes :

i) assigning a root name to a root thread,

ii) maintaining a thread-bound index structure for holding for each thread a corresponding index counter which is atomically incremented upon thread creation, and

iii) upon creation of a child thread assigning a name including a prefix indicative of a name of a respective parent thread and a suffix indicative of an index counter of the respective parent thread.

15. (previously presented) The method according to Claim 11, wherein operation **(a)(ii)** includes:

i) attempting a bipartite matching between the threads and the traces such that every thread has a trace which contains what the thread printed, and

ii) if said bipartite matching is not possible, then stopping the program so as to avoid executing debugger commands embedded in each of the traces at the wrong time.

16. (original) The method according to Claim 11, further including providing a mechanism for manually or automatically bypassing step **(a)(ii)** so that traces are created in respect of only a subset of the threads.

17. (previously presented) The method according to Claim 11, wherein operation **(b)** includes:

i) reading the modified trace file in respect of local views of the threads only, so as to avoid a need for synchronizing break-points in said multiple threads.

18. (previously presented) A computer program storage device readable by machine, tangibly embodying a program of instructions executable by the machine to perform a method for automatically invoking at least one predetermined debugger command at a desired location of a single thread of a program containing at least one thread, said method comprising:

(a) embedding within said program thread at said desired location thereof a utility which reads a trace file in which said at least one predetermined debugger command has been previously embedded; and

(b) running the program so that on reaching said desired location, the utility is invoked for reading said trace file and invoking said at least one predetermined debugger command.

19. (previously presented) A computer program product comprising a computer useable medium having computer readable program code embodied therein for automatically invoking at least one predetermined debugger command at a desired location of a single thread of a program containing at least one thread, said computer program product comprising:

computer readable program code for causing the computer to embed within said program thread at said desired location thereof a utility which reads a trace file in which said at least one predetermined debugger command has been previously embedded; and

computer readable program code for causing the computer to run the program so that on reaching said desired location, the utility is invoked for reading said trace file and invoking said at least one predetermined debugger command.

20. (previously presented) A computer program storage device readable by machine, tangibly embodying a program of instructions executable by the machine to perform a method for automatically invoking at least one predetermined debugger command at a desired location of a specific thread of a program containing at least one thread, said method comprising:

(a) embedding within the specific thread of the program at said desired location thereof a utility which:

i) checks whether a trace file exists on entry to the utility,

ii) if the trace file does not exist on entry to the utility, creates the trace file and writes a traced value of at least one variable thereto at a desired location of the program, and

iii) if the trace file does exist on entry to the utility, compares a current value of the least one variable with a respective line in the trace file and if they are different invokes a debugger so as to execute a debugger command embedded in the trace file in place of the traced value; and

(b) running the program so that on reaching said desired location, the utility is invoked for reading a modified trace file readable by the program wherein at least one traced value is replaced or augmented by said debugger command.

21. (previously presented) A computer program product comprising a computer useable medium having computer readable program code embodied therein for automatically invoking at least one predetermined debugger command at a desired location of a specific thread of a program containing at least one thread, said computer program product comprising:

computer readable program code for causing the computer to embed within the specific thread of the program at said desired location thereof a utility which reads a trace file in which said at least one predetermined debugger command has been previously embedded,

computer readable program code for causing the computer to check whether a trace file exists,

computer readable program code for causing the computer to create the trace file if the trace file does not exist on entry to the utility, and to write a traced value of at least one variable thereto at a desired location of the program, and

computer readable program code for causing the computer to compare a current value of the least one variable with a respective line in the trace file if the trace file does exist on entry to the utility, and if they are different to invoke a debugger so as to execute a debugger command embedded in the trace file in place of the traced value; and computer readable program code for causing the computer to run the program so that on reaching said desired location, the utility is invoked for reading a modified trace file readable by the program wherein at least one traced value is replaced or augmented by said debugger command.

22. (previously presented) A computer program storage device readable by machine, tangibly embodying a program of instructions executable by the machine to perform a method steps for automatically invoking at least one predetermined debugger command at a desired location of a specific thread of a program containing at least one thread, said method comprising:

(a) checking whether a trace file exists on entry to the utility,

(b) if the trace file does not exist on entry to the utility, creating the trace file and writing a traced value of at least one variable thereto at a desired location of the program, and

(c) if the trace file does exist on entry to the utility, comparing a current value of said at least one variable with a respective line in the trace file and if they are different invoking a debugger so as to execute a debugger command embedded in the trace file in place of the traced value.

23. (previously presented) A computer program product comprising a computer useable medium having computer readable program code embodied therein for automatically invoking at least one predetermined debugger command at a desired location of a specific thread of a program containing at least one thread, said computer program product comprising:

computer readable program code for causing the computer to check whether a trace file exists on entry to the utility,

computer readable program code for causing the computer to create the trace file if the trace file does not exist on entry to the utility, and to write a traced value of at least one variable thereto at a desired location of the program, and

B

computer readable program code for causing the computer to compare a current value of the at least one variable with a respective line in the trace file if the trace file does exist on entry to the utility, and if they are different to invoke a debugger so as to execute a debugger command embedded in the trace file in place of the traced value.

24. (previously presented) A computer-implemented system for automatically invoking at least one predetermined debugger command at a desired location of a single thread of a program containing at least one thread, said system comprising:

a code modifier for embedding within said single thread at said desired location thereof a utility which reads a trace file in which said at least one predetermined debugger command has been previously embedded, and

a processor for invoking said utility for reading said trace file during running of the program upon reaching said desired location and invoking said at least one predetermined debugger command.

25. (previously presented) The system according to Claim 24, further including:

a file management system coupled to the processor and responsive to said utility for checking whether a trace file exists on entry to the utility, and for creating the trace file if it does not exist on entry to the utility,

a file modifier coupled to the file management system and responsive to the trace file being created for writing to the trace file a traced value of at least one variable at said desired location of the program, and

a comparator coupled to the processor for comparing a current value of the least one variable with a respective line in the trace file and if they are different construing the respective line in the trace file as a debugger command and invoking a debugger so as to execute the debugger command.

B

26. (original) The system according to Claim 24, wherein the debugger is adapted to attach itself to a predetermined debugger command, which halts execution of the program and shows a state of the program at that time.

27. (original) The system according to Claim 24, wherein the program is multi-threaded and there is further included a replay mechanism coupled to the processor for rerunning the program with identical interleaving as far as instrumentation statements are concerned.

28. (previously presented) A computer-implemented system for automatically invoking at least one predetermined debugger command at a desired location of a specific thread of a program containing at least one thread, said system comprising:

a code modifier for embedding within said single thread at said desired location thereof a utility which reads a trace file in which said at least one predetermined debugger command has been previously embedded,

a file management system coupled to the processor and responsive to said utility for checking whether a trace file exists on entry to the utility, and for creating the trace file if it does not exist on entry to the utility,

a file modifier coupled to the file management system and responsive to the trace file being created for writing to trace file a traced value of at least one variable at said desired location of the program,

a processor for running the program so that on reaching said desired location, the utility is invoked for reading a modified trace file readable by the program wherein at least one traced value is replaced or augmented by said debugger command, and

a comparator coupled to the processor for comparing a current value of the least one variable with a respective line in the trace file and if they are different construing the respective line in the trace file as a debugger command and invoking a debugger so as to execute the debugger command.

29. (original) The system according to Claim 28, wherein the program is multi-threaded and there is further included a replay mechanism coupled to the processor for rerunning the program with identical interleaving as far as instrumentation statements are concerned.

30. (original) The system according to Claim 28, wherein the program includes multiple threads and the file management system is adapted to create for each thread a respective trace file having a name which is uniquely defined by a name of the respective thread, thereby allowing debugger commands embedded in any of the trace files to be executed during a respective one of the threads.

31. (original) The system according to Claim 30, wherein the file management system is responsive to a predetermined execution-independent naming scheme for automatically naming said trace files.

32. (original) The system according to Claim 31, wherein the file management system includes:

an assignment unit for assigning a root name to a root thread, and

a thread-bound index structure for holding for each thread a corresponding index counter which is atomically incremented upon thread creation;

said assignment unit being responsive to creation of a child thread for assigning a name including a prefix indicative of a name of a respective parent thread and a suffix indicative of an index counter of the respective parent thread.

33. (original) The system according to Claim 32, wherein the assignment unit is responsive to no consistent naming being possible for attempting a bipartite matching between the threads and the traces such that every thread has a trace which contains what the thread printed, and for stopping the program so as to avoid executing debugger commands embedded in each of the traces at the wrong time if said bipartite matching is not possible.

34. (original) The system according to Claim 28, further including a bypass mechanism coupled to the file modifier for allowing creation of the trace file to be manually or automatically bypassed so that traces are created in respect of only a subset of the threads.

35. (original) The system according to Claim 28, wherein the processor is adapted to read the modified trace file in respect of local views of the threads only, so as to avoid a need for synchronizing break-points in said multiple threads.